A General Purpose Configurable Navigation Controller for Micro Aerial Multirotor Vehicles

Jesús Pestana , Ignacio Mellado-Bataller , Changhong Fu , José Luis Sanchez-Lopez , Iván Fernando Mondragón , and Pascual Campoy

I. INTRODUCTION

Micro Air Vehicles are proving to be a reliable platform for civilian applications because they are easy to deploy on site, and they are cost-effective compared to traditional options. Moreover, their little size and weight are important factors that reduce the risk associated to employing flying platforms in the air-space near populated areas. Different types of platform, such as fixed-wing and rotary-wing micro Unmanned Aerial Vehicles (mUAVs), can be the most advantageous depending on the operation requirements of the task at hand, allowing the acquisition of sensor data, images and video streaming from the air. Some examples of applications where mUAVs are being applied are aerial mapping, inspection tasks in construction sites, post-disaster damage assessment for insurance estimations and agriculture, environmental and wildlife monitoring.

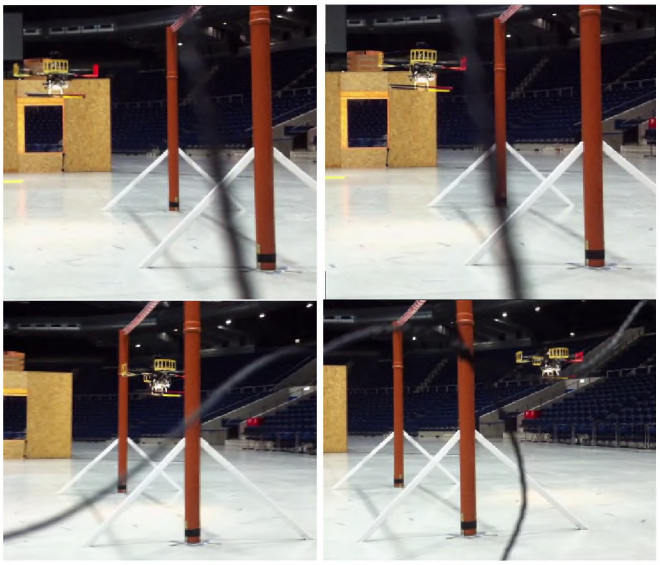

Fig. 1. The Asctec Pelican quadrotor, equipped as explained in Sec. III, is shown flying in autonomous mode on the IMAV 2012 competition with an implementation of the controller presented in this paper. An EKF performs state estimation on the odometry measurements, and its estimation is fused with the position estimations from a Montecarlo Localization algorithm that is processing the Laser Range Finder readings. The quadrotor also had to be able to fly safely on areas where no obstacles were in range of the laser sensor. In this image sequence the quadrotor passes through a pair of poles using the laser sensor to estimate its position with respect to the poles. This flight gained the two IMAV 2012 awards stated in the abstract, see http://vision4uav.com/?q=IMAV12 . The full video of the flight is available online on http://vision4uav.com/?q=node/323 .

The presented research is focused on multirotor platforms, that are capable of flying in cluttered areas and hover safely in position. In order to obtain a semi-autonomous flying platform that can be commanded with high-level commands by an operator, the state estimation and the control problem

must be addressed. The presented work has been tested experimentally with several multirotors: the AR Drone, the Asctec Pelican and the LinkQuad quadrotors. The code related to the presented work, which has been developed in our research group, is available in two open-source projects in GitHub[1] & [2], refer to section III-B for more information. Note that the work is still on-going and not all the control modes are currently available for the LinkQuad quadrotor.

II. Related Work

The following cited work has shown that the navigation control loops must be designed taking into account the non-linear dynamics of the multirotor, so that the control actions are coupled. If the control laws are well designed the multirotor can perform smooth trajectories in position coordinates, while orientating its yaw heading in any required direction.

Several labs have used a sub-millimeter accurate Vicon motion tracking system[3] to separate the control and the state estimation problems. Researchers at the GRASP lab of the University of Pennsylvania, and at the ETH - IDSC - Flying Machine Arena in the ETHZ, have been able to execute precise trajectory following[4], [5], to perform aggresive maneuvers[5], [6], and to perform collaborative tasks that require synchronization among the flying vehicles[7], [8]. Relying on Vicon motion capture systems has simplified the research problem, which has shown that state estimation is the key to enabling many autonomous applications.

On the other hand, there are research groups that have shown successful autonomous multirotor navigation capabilities using only GPS positioning data. The STARMAC project[9], from Stanford University has shown several experimental tests on outdoors ranging from trajectory tracking tasks[10] to performing flips and stall-turn maneuvers[11], [12].

Other groups have increased the situational awareness of the mUAVs using optical flow sensors and cooperative robots. For instance, a ground vehicle can estimate the position and attitude of the mUAV[13]. These two research works [14], [15] are focused on navigation in corridors and obstacle collision avoidance strategies using multiple optical flow sensors. In outdoors navigation, optical flow sensors were used in a fixed-wing mUAV in[16] to avoid the collision with trees and other obstacles in a GPS waypoint trajectory.

III. System Overview

The presented research is a continuation on previous work by the same authors[17], [18], that was implemented to participate on the IMAV 2012 indoors dynamics challenge[19]. Images from the flight that gained the awards of the IMAV 2012 competition are shown in Fig. 1, where the Asctec Pelican was controlled by the software architecture presented in this paper. In this case the localization problem was engaged using an Extended Kalman Filter (EKF) to fuse the odometry measurements, and a Particle Filter (PF) with a known map of the environment. The PF processed the Laser Range Finder readings providing estimated positions with

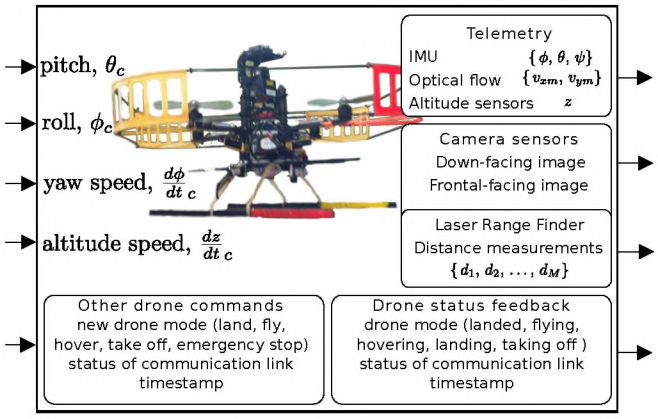

Fig. 2. Pelican on CVG Framework for interfacing with MAVs[1], user point of view. The Pelican offers several feedback channels: the telemetry package with IMU attitude, horizontal body speed and altitude data, and camera feedback channels where each camera is identified with an ID value. The drone modes are replicated from those available on the AR Drone.

respect to the pylons of the challenge[19]. The quadrotor also had to be able to fly safely on areas where no obstacles were in range of the laser sensor, where only the EKF provides the necessary feedback to the controller. The focus of this paper is to further researh on safe navigation based on unperfect odometry measurements, such as on-board optical flow measurements, and no position measurements are available.

This paper presents a navigation system architecture which controller can be configured depending on the limitations that are imposed by the kinematic capabilities of the vehicle, by the measurement range of the speed estimators, or by precision requirements of the task at hand. The result is an architecture that was experimentally tested on GPS-denied environments, and that can be configured depending on the requirements of each phase of a task. This allows to have a setup for fast trajectory following, and another to soften the control laws and make the vehicle navigate more precisely and slowly whenever necessary.

A. Hardware Architecture

The implemented hardware architecture is inspired on the AR Drone quadrotor, which capabilities are described in[20]. An Asctec Pelican quadrotor was equipped with a camera and an on-board computer to obtain a similar setup. The advantages of using the Asctec Pelican are that all the algorithms can run on the on-board computer, and that the vehicle can be equipped with more sensors such as cameras and Laser Range Finders.

The Asctec Pelican shown in Figure 2 is equipped with an autopilot board that stabilizes the vehicle using information from GPS (only when outdoors), IMU, pressure altimeter and magnetometer fused using a Kalman Filter. This controller is embedded, closed, unmodifiable but gains are tunable. This quadrotor has also been equipped with a series of complementary sensors: a Hokuyo Scanning Laser Range Finder for horizontal depth mapping, a sonar altitude sensor for height estimation improvement and two cameras, one

of them a downward looking sensor for velocity estimation based on optical flow. The optical flow is calculated using the OpenCV implementation of the pyramidal Lucas-Kanade algorithm for some image features selected by the Shi-Tomasi method. The state estimation and control processing are exeuted onboard in an Atom Board, which has a dual core Atom 1.6 GHz processor with 1 GB of RAM.

B. Software Architecture

The AR Drone and the Asctec Pelican were both accomodated to be compatible with the CVG Framework for interfacing with MAVs. This framework, also called "MAVwork", was developed by our research group; it is an open-source project hosted on GitHub that can be accessed in [1], and its technical details can be found on [21]. One of the advantages of the framework is that it offers a common type of interface for every multirotor, allowing other software modules, such as the controller, to be compatible with multiple multirotors. The MAVwork interface has been designed to be similar to the one offered by the AR Drone, as this quadrotor has proven to be easy and reliable to setup for experimental tests. Among other advantages, MAVwork offers native take-off, landing and hovering flying modes; along with altitude hold capability. Fig. 2 summarizes the interface functions that the API and drone object offer to the developer.

The different components of the control and state estimation modules are shown in Fig. 3. An Extended Kalman Filter is used as data fusion algorithm to estimate the position and speed of the quadrotor. These estimates are then fed to the high-level (HL) controller that obtains the position, speed and feedforward commands that are given to the mid-level controller. The planning process that takes place in the HL controller uses a simple kinematic model of the multirotor that is specified by a little set of configurable parameters. The mid-level (ML) controller calculates and sends the commands to the CVG framework, which acts as interface between the controller and the quadrotor. Several saturation rules are used in order to perform the trajectory following task under certain safety conditions. The HL and ML controllers are further described in section IV. The state estimation and controller algorithms are available on an open-source project called "Multirotor Controller for MAVwork" hosted on GitHub that can be accessed in [2]. Its technical details are explained in the Msc. Thesis [22] and on the present paper.

The navigation of the multirotor is stabilized using odometry-based position estimates and for that reason the boundaries where the quality of these estimates is lost must be determined. For instance, the optical flow algorithm that is calculated on the down-facing images is affected by the texture and reflectiveness of the floor surface, and on the lighting conditions of the environment. The size of the floor texture details and the computing power of the on-board computer set a maximum speed boundary on the speed estimation. The maximum speed that could be reliably measured in the indoors playground where most of the tests

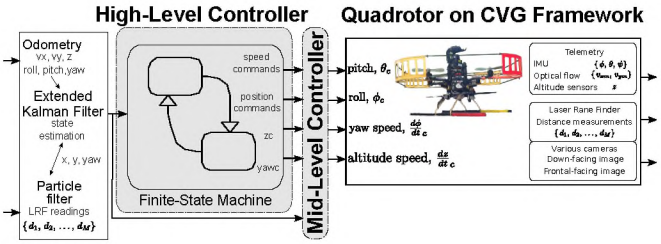

Fig. 3. General software architecture of the navigation controller. The State Estimation is used by the High-Level controller to determine optimized reachable speed commands to the mid-level controller, which sends commands to the drone through the CVG Framework[1], [21]

where carried out was about 0.5-0.75 m/s, and so, this boundary was used in our indoors tests.

The presented software architecture has been implemented on the AR Drone and Astec Pelican quadrotors, and will be applied in the future to other multirotor platforms, more specifically to a LinkQuad[23] quadrotor and an OktoKopter from Mikrokopter[24]. The usage of a common software architecture increases code quality shareability and enables for different research lines to work in a cohesive way towards the objectives of our group.

IV. NAVIGATION CONTROLLER

This section introduces the navigation controller that uses a set of configurable parameters to adapt to different multirotors and sensor limitations. The parameters are inspired by the STARMAC project work described in[10]. First a simple multirotor point mass kinematic model is described. Then the high level controller state-machine is introduced with the description of the configurable parameters that are derived from the model. The final part of the section introduces the middle-level controller.

A. Multirotor Point Mass Kinematic Model

A multirotor is a flying vehicle that can exert force in any direction and thus can be simplified as a point mass moving in the 3D space. The vehicle tilts to exert force in the horizontal plane, and changes the average propeller speeds to control the force in the altitude coordinate. A maximum speed is also set by various reasons, for instance, due to the aerodynamic friction of the movement through the air. It is also important to take into account that the multirotor is commanded using feedback loop control laws that produce a response that can be characterized by a little set of parameters. The speed planifier can take advantage of a simple kinematic model built upon these facts in order to take decisions.

The body frame, $\{X_m, Y_m, Z_m\}$, of the vehicle, shown in Fig. 4, is described as follows. The $\{X_m, Y_m\}$ axis are horizontal and the Z_m axis points down to the floor. The X_m axis points to the front of the vehicle and the Y_m axis points to the right obtaining a orthonormal right-handed reference frame. The euler angles are denoted $\{\phi, \text{roll}\}$, $\{\theta, \text{pitch}\}$ and $\{\psi, \text{yaw}\}$, and they define the rotation between the body frame, $\{X_m, Y_m, Z_m\}$, and the world frame, $\{X, Y, Z\}$.

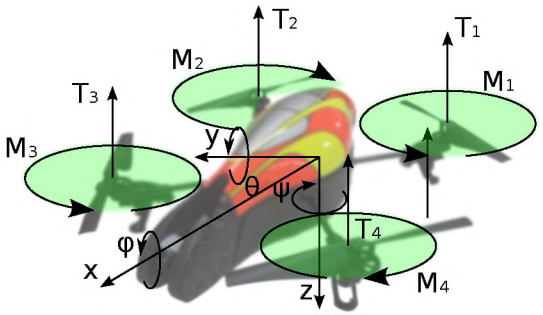

Fig. 4. Free body diagram of a quadrotor. The four propellers, and each of their performed thrust T_i and torque M_i are labeled 1-4. The euler angles are denoted $\{\phi$,roll$\}$, $\{\theta$,pitch$\}$ and $\{\psi$,yaw$\}$

The quadrotor rigid body dynamics model, see Eq. 1 and Fig. 4, is easy to infer from physical laws and is explained in multiple articles [10], [5]:

$$\begin{aligned}
I_x\ddot{\phi} &= \dot{\psi}\,\dot{\theta}\,(I_y - I_z) + l\,(T_1 + T_2 - T_3 - T_4)\\
I_y\ddot{\theta} &= \dot{\phi}\,\dot{\psi}\,(I_z - I_x) + l\,(T_1 + T_4 - T_2 - T_3)\\
I_z\ddot{\psi} &= \dot{\theta}\,\dot{\phi}\,(I_x - I_y) + \textstyle\sum_{i=1}^{4} M_i\\
m\ddot{x} &= (-sin\phi\,sin\psi - cos\phi\,sin\theta\,cos\psi)\textstyle\sum_{i=1}^{4} T_i - K_{fr}\dot{x}^2\\
m\ddot{y} &= (-cos\phi\,sin\theta\,sin\psi + sin\phi\,cos\psi)\textstyle\sum_{i=1}^{4} T_i - K_{fr}\dot{y}^2\\
m\ddot{z} &= mg - cos\theta\,cos\phi\textstyle\sum_{i=1}^{4} T_i
\end{aligned} \tag{1}$$

Where the following the following variables have been introduced: the position of the quadrotor with respect to the world frame is denoted by the $\{x, y, z\}$ coordinates, the attitude of the vehicle is represented by the yaw ψ, pitch θ and roll ϕ euler angles; the quadrotor rigid body is characterized by its mass m and its three principal mass moments of inertia $\{I_x,\ I_y,\ I_z\}$, each propeller i generates a thrust T_i and a heading torque M_i, K_{fr} is an aerodynamic friction constant and the l constant is the arm between each pair of thrusts, in this case the distance between the propellers 1 and 2, as denoted in Fig. 4.

The following control loops are usually implemented inside the autopilot board of the multirotor: the attitude (roll, ϕ, pitch, θ, and yaw, ψ) control loops and altitude, z, control loop. The autopilot boards usually accept roll, pitch, yaw speed and altitude speed references. So that the saturation bounds for these variables is set by the autopilot board capabilities. Thus, it is only required to further develop the equations of mouvement in the horizontal plane.

If the flight is performed at constant altitude then, $\sum_{i=1}^{4} T_i \approx mg$, and taking the approximation to low roll and pitch angles, then the equations that define the mouvement in the horizontal plane are derived from Eqs. 1:

$$\begin{aligned}
m\ddot{x} &= (-\theta\,cos\psi - \phi\,sin\psi)\,mg - K_{fr}\dot{x}^2\\
m\ddot{y} &= (-\theta\,sin\psi + \phi\,cos\psi)\,mg - K_{fr}\dot{y}^2
\end{aligned} \tag{2}$$

Two additional attitude variables are defined $\{\phi_v$,virtual roll$\}$, $\{\theta_v$,virtual pitch$\}$, which control de position of the vehicle in the horizontal $\{X, Y\}$ plane. They are related to the actual $\{\phi, \theta\}$ through the yaw (ψ) angle as follows:

$$\begin{bmatrix}\theta_v\\ \phi_v\end{bmatrix} = \begin{bmatrix}\cos(\psi) & \sin(\psi)\\ -\sin(\psi) & \cos(\psi)\end{bmatrix}\begin{bmatrix}\theta\\ \phi\end{bmatrix} \tag{3}$$

The attitude controller ($t_r \approx$ 200-300 ms) is significantly faster than the horizontal speed loop ($t_r \approx$ 1-2 s). This fact allows to correct the coupling introduced by the yaw commands in the horizontal speed loops using the decoupling law, defined by Eq. 3. The reference frame change defined by Eq. 3 is included in the controller, so that the $\{\phi_v, \theta_v\}$ can be used for the controller design.

The dynamic model of the multirotor is much simpler when expressed using the virtual angles and a maximum value for the quadrotor acceleration saturation can be infered from Eqs. 2 & 3 as follows:

$$\ddot{x} \approx -g\,\theta_v - (K_{fr}/m)\,\dot{x}^2 \rightarrow |\ddot{x}| \leq g\,|\theta_{v\,max}| - (K_{fr}/m)\,\dot{x}^2 \tag{4}$$

$$\ddot{y} \approx g\,\phi_v - (K_{fr}/m)\,\dot{y}^2 \rightarrow |\ddot{y}| \leq g\,|\phi_{v\,max}| - (K_{fr}/m)\,\dot{y}^2 \tag{5}$$

The maximum horizontal speed of the vehicle, $v_{xy\,max}$, is physically limited by the aerodynamic friction. Thus, the maximum quadrotor speed is expressed by Eq. 7 and the following model:

$$(\dot{x}^2 + \dot{y}^2) \leq v_{xy\,max}^2 \tag{6}$$

The altitude acceleration can be modeled similarly, but the AR Drone only accepts altitude speed commands; which is the reason to constrain the altitude speed and not its acceleration:

$$|\dot{z}| \leq \frac{dz}{dt}_{c\,max} \tag{7}$$

The point mass model has to take into account that the vehicle is being commanded using feedback control laws, and that the controller will not be able to push the vehicle capabilities to the physical limits stated above. Thus the following set of parameters is selected based on this fact and on the previous considerations:

- The combination of Eqs. 7 & 6 set a maximum speed on any direction.
- The speed response of the vehicle can be characterized by the response time of the speed control loop $t_{rv\,max}$
- The maximum acceleration is physically limited by Eqs. 4 & 5, and can be further determined by experimental tests. But the actual accelerations obtained during trajectotry control depend on the performances of the controller and on the smoothness of the speed references set to the speed control loop. Thus, the maximum acceleration parameters in the horizontal plane are derived $\{a_{x\,max}, a_{y\,max}\}$.
- The trajectory around checkpoints can be approximately modeled by a sector of circumference thus requiring a centripetal acceleration that is limited by

$$a_{cxy} \leq g\{\ \theta_{v\,max},\ \phi_{v\,max}\} = \frac{v_t^2}{R_c} \tag{8}$$

where v_t and R_c are the average speed and curve radius during the turn.

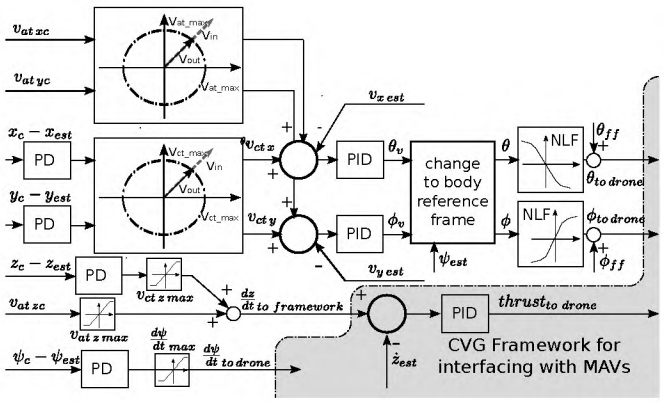

Fig. 5. The middle-level controller architecture is a cascade controller, which consists of an inner speed loop and an outer position loop. The reference change block, which implements Eq. 3 ensures that the roll and pitch commands are decoupled from the yaw heading. This allows for the yaw command to be set independently from the position and speed references. Thus, separating the left part of the diagram which is expressed in world coordinates, from the right part that is calculated in body frame coordinates.

B. Middle-Level Controller

The mid-level controller is shown in Fig.5 and it is inpired on previous research from other groups[4], [5], [10]. Its general architecture is a cascade controller consisting of an inner speed loop and an outer position loop. But it also includes control laws to take into account the overall system capabilities. Some characteristics of the control architecture decipted in Fig. 5 are:

- The position controller loop only commands achievable speeds to the speed controller loop, see left block in Fig. 5. The reference (feedforward) speed along the trajectory is saturated to $v_{at\,max}$ and $v_{at\,z\,max}$; and the cross-track reference speed is saturated to $v_{ct\,max}$ and $v_{ct\,z\,max}$. Both speed limits are set so that $v_{at\,max}+v_{ct\,max} \leq v_{xy\,max}$, $v_{at\,z\,max}+v_{ct\,z\,max} \leq v_{z\,max}$.
- The planned along-track velocity $\{v_{at\,xc}, v_{at\,yc}, v_{at\,zc}\}$ is lower than the maximum velocity, thus, giving the relative position controller a speed margin to work on.
- The reference change block, which implements Eq. 3 ensures that the roll and pitch commands are decoupled from the yaw heading.
- The aerodynamic friction is partially linearized, in the NLF blocks, using the inverse of the identified aerodynamic friction.
- In the PID modules the derivate action is filtered to limit noise amplification and the integral action features anti-windup saturation.
- The variables $\{\theta_{ff}, \psi_{ff}\}$ are feed-forward commands that are calculated by the state machine using the planned acceleration and Eqs. 4 & 5 without considering the aerodynamic friction.

C. High-Level Controller - State-Machine

The HL controller is implemented using a Finite State Machine (FSM), which uses the state estimation from the EKF to calculate the mid-level controller references. The FSM has three states corresponding to three control strategies: hover in a desired position, follow a straight segment of the trajectory and turn to head to the next trajectory segment. The work of the FSM is to navigate along a waypoint trajectory, and it can be summarized as the repetition of the following steps:

1) Follow the current straight segment, accelerating at first, and slowing down before reaching the next turn,
2) Perform the turn, controlling the speed direction to achieve a soft alignment with the next straight segment.

Every time the controller output is calculated the FSM calculates the planned speed at the current position $v_{plan}(s)$:

1) The speed, $\{V_{turn\,i}\}$, inner turn angle at checkpoint, $\{\alpha_i\}$, and radius, $\{R_{turn\,i}\}$, during each turn are precalculated each time the reference trajectory is changed, either because the whole trajectory is changed or because a waypoint was added. The radius is calculated so that the trajectory passes at a distance $R_{conf}/2$ from the checkpoint, where R_{conf} is the maximum distance at which the checkpoint is considered to be reached. The planned speed for each turn is selected either according to Eq. 8 either considering that the turn requires a stall-turn or turn-around maneuver, $v_{stall\,turn}$, which corresponds to a planned speed near zero.
2) The algorithm explained in Sec. IV-D is used to calculate the planned speed, $v_{plan}(s)$, at the current position.

The FSM has also two additional operation modes where the quadrotor is controlled either only in position control, or in speed control. These can be used by the task scheduler to perform certain tasks.

The middle-level (ML) controller, see Fig. 3, receives position, $\{x_c, y_c, z_c\}$, yaw heading, ψ_c, speed, $\{v_{at\,xc}, v_{at\,yc}, v_{at\,zc}\}$, and tilt feed-forward, $\{\theta_{ff}, \psi_{ff}\}$, commands from the FSM, as shown in Fig. 5. The presented controller, which is inspired on those presented in [10], [5], allows to use single mid-level architecture for the three control modes: trajectory, position and speed modes. These references to the ML controller are calculated by the FSM in such a way that the relative position and speed commands are orthogonal:

- The position references $[x_c, y_c, z_c]$ are caltulated projecting the estimated position $[\,x_{est},\ y_{est},\ z_{est}]$ onto the reference trajectory.
- The along-track speed commands, $[v_{at\,xc}, v_{at\,yc}, v_{at\,zc}]$, are parallel to the trajectory $v_{plan}(s)\,\vec{u}_{tangent}$.
- The along-track speed commands, $[v_{at\,xc}, v_{at\,yc}, v_{at\,zc}]$, are derivated and smoothed using a low-pass filter to obtain an acceleration command based on Eqs. 4 & 5, so that the feed-forward attitude references, $\{\theta_{ff}, \phi_{ff}\}$, are calculated as follows:

$$\{\theta_{ff}, \phi_{ff}\} = \left\{-asin\left(\frac{\ddot{x}_m}{g}\right), asin\left(\frac{\ddot{y}_m}{g}\right)\right\}$$

D. Speed Planner Algorithm

The planned speed depends on the current speed and position of the multirotor, and on the previous and subsequent trajectory waypoints. The speed planner uses a

uniform acceleration motion model to calculate both: the acceleration profile from the previous trajectory waypoints, and the deceleration profile to perform the following turns succesfully. In addition to this model the speed loop response is modeled taking into account its response time.

The algorithm calculates the following distances for each of the neighbouring waypoints:

1) Δd_{chki}, the distance to waypoint i from the current estimated quadrotor position.
2) $\Delta d_{turni} = (R_{turni} + \frac{R_{conf}}{2}) sin(\alpha_i/2)$, the distance to waypoint i when the turn is to be started; where R_{conf} is the waypoint maximum clearance distance, $\{R_{turni}\}$ is the planned turn radius at waypoint i and $\{\alpha_i\}$ is the inner turn angle at waypoint i.
3) $\Delta d_{trvi} = t_{rvmax} \sqrt{v_{xest}^2 + v_{yest}^2}$, the distance required by the speed loop to slow down, where $\{v_{xest}, v_{yest}\}$ is the current estimated velocity and t_{rvmax} is the speed loop response time.

Then the optimal speed for the neighbouring checkpoints, v_{plan_i}, is calculated depending on the sign of $\Delta d = (\Delta d_{chki} - \Delta d_{turni} - \Delta d_{trvi})$. If $\Delta d \geq 0$ then $v_{plan_i} = \sqrt{V_{turni}^2 + 2 a_{xymax} \Delta d}$, else if $\Delta d < 0$ then $v_{plan_i} = V_{turni}$. Where $\{V_{turni}\}$ is the planned speed at turn i. Finally the planned speed at the current position, $v_{plan}(s)$, is calculated as the minimum of the precaltulated optimal speeds $v_{plan}(s) = arg\,min_i\,(v_{plan_i})$. The coordinate s denotes the position of the quadrotor along the trajectory, and the expression $v_{plan}(s)$ highlights that the planned speed depends only on s and the following set of configurable parameters.

E. Configurable Parameters Set

From the prior explanation the set of configurable parameters is derived:

- $\{v_{atmax}, v_{ctmax}, v_{atzmax}, v_{ctzmax}\}$, are the saturation speed limits used inside the middle-level controller that are imposed by either the kinematic capabilities of the vehicle, or the measurement range of the onboard sensors or the precision requirements of the task at hand.
- $\{R_{conf}, v_{stallturn}, t_{rvmax}, a_{xymax}\}$, are the parameters that are used to determine the planned speed at each turn, $\{V_{turni}\}$, and to calculate the planned speed, $v_{plan}(s)$, at every controller step iteration.

V. EXPERIMENTAL RESULTS

This section describes four separate experimental flights that show the navigation capabilities of the presented system architecture. All the flights are trajectory tracking tests in autonomous navigation carried out in an indoors espacious room, simulating a GPS-denied situation. Thus, the GPS signal is not used in the estimation algorithms. One of the tests consists on the Pelican following a trajectory in a vertical XZ plane, see Fig. 6. There is a second test performed by this quadrotor following a square horizontal trajectory shown on Fig. 7. The last two tests were done using the AR Drone following the horizontal eight-shaped trajectory shown in Figs. 8 & 9. A summary of the controller performances in these tests is shown in Table I.

The Pelican was configured with the following parameters values. $\{v_{atmax}, v_{ctmax}, v_{atzmax}, v_{ctzmax}\}$ where fixed to 0.5 m/s. and the horizontal speed was constrained with an additional saturation to 0.70 m/s. This limitation ensured that the speed reference for the speed control loop was inside the reliable measurement range of our optical flow implementation. The FSM parameters were: $R_{conf} = 0.30\ m$, $v_{stallturn} = 0.30\ m/s$, t_{rvmax} was set to 0 s because the planned speed was always lower than 0.7 m/s (this parameter does not give an advantage in this situation), and the maximum acceleration a_{xymax} was set to 0.5 m/s^2.

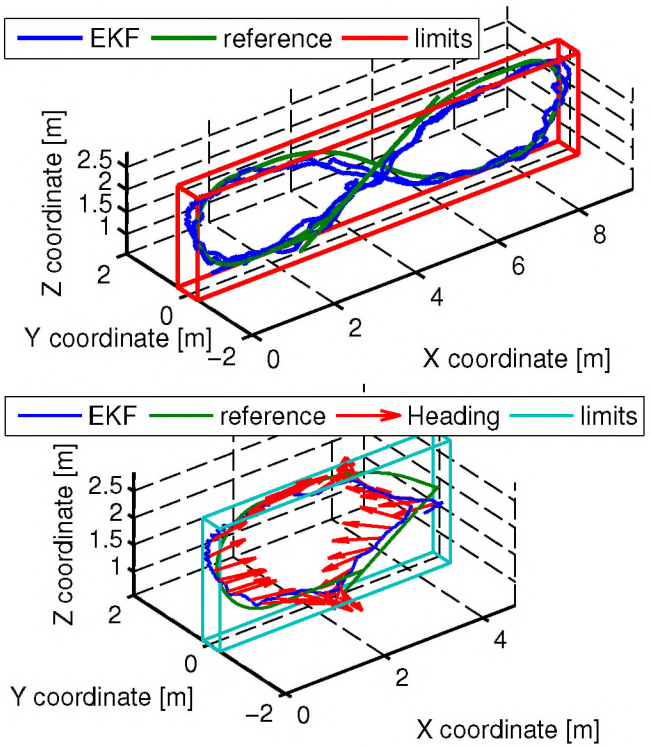

Fig. 6. Asctec Pelican test on a vertical eight-shape trajectory, where two laps were completed. The graph above shows the trajectory in the XYZ space, and the bounding box graph ('limits' on the legend) shows that the Y error was kept low. A video of the test is available on http://vision4uav.com/?q=node/338 . During the second lap the yaw reference was changed repeatedly without affecting the trajectory tracking.

The vertical flight of the Pelican on the XZ plane shown in Fig. 6 consisted on two laps. The mean and maximum tracking errors during the test were 0.15 m and 0.38 m correspondingly. The vehicle navigated at a mean and maximum speeds of 0.42 m/s and 0.7 m/s. During half of the flight the vehicle was commanded to change its yaw heading, without affecting the trajectory tracking objectives.

The horizontal flight of the Pelican shown in Fig. 7 was carried out setting a constant yaw heading reference so that the results could be compared with those of the previous test, shown on Fig. 6. The mean and maximum tracking errors during the test were 0.09 m and 0.27 m correspondingly.

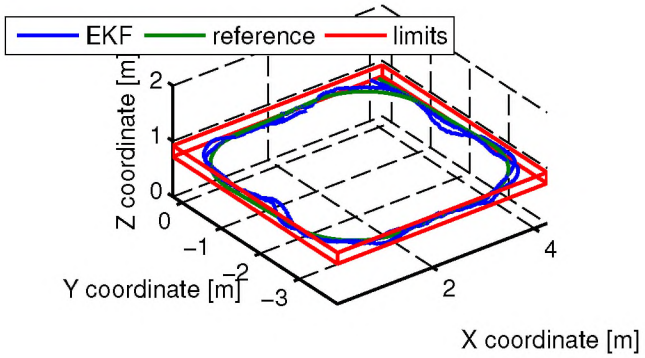

Fig. 7. Asctec Pelican test on horizontal square trajectory. The quadrotor performed two turns carrying out the trajectory tracking task. The altitude hold worked successfully during this test, and the yaw heading reference was kept constant.

The Pelican navigated at a mean and maximum speeds were 0.51 m/s and 0.60 m/s. The trajectory controller features similar performances on both types of trajectory.

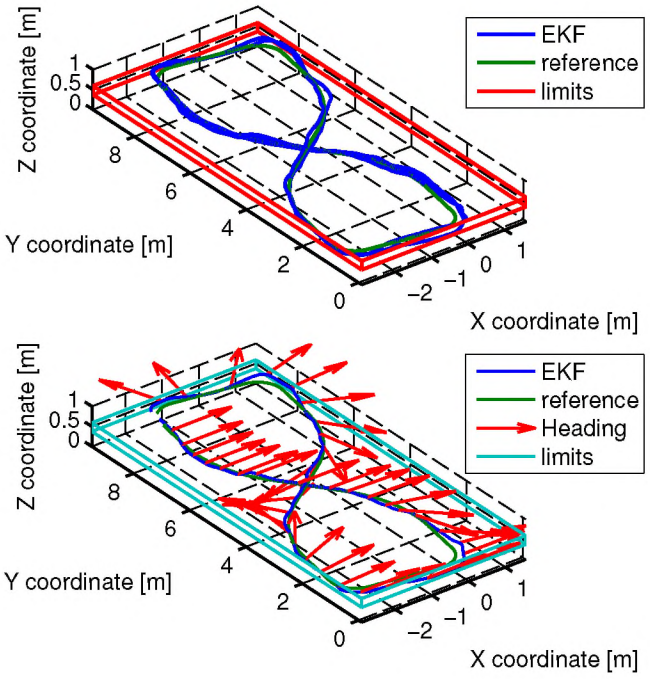

Fig. 8. AR Drone following a horizontal eight-shape trajectory. The first graph shows the whole flight consisting of two and a half laps. In the second graph only the lap where yaw heading reference was changed is shown. This rotation does not affect much the trajectory tracking, however, small oscillations around the trajectory reference can be observed. A video of the test is available in http://vision4uav.com/?q=node/338 .

The AR Drone was configured with the following parameters values. The maximum horizontal speed was set to 2 m/s, where $v_{at\,max} = 1.4$ m/s, $v_{ct\,max} = 0.8$ m/s. The FSM parameters were set to $R_{conf} = 0.50\ m$, $v_{stall\,turn} = 0.25\ m/s$, $t_{rv\,max} = 1.7$ s, and the maximum acceleration to $a_{xy\,max} = 0.4$ m/s^2. The configuration parameters were selected to work on trajectories made up of short segments, with the objective of optimizing speed while keeping low tracking error.

Both flights shown in Figs. 8 & 9, were performed

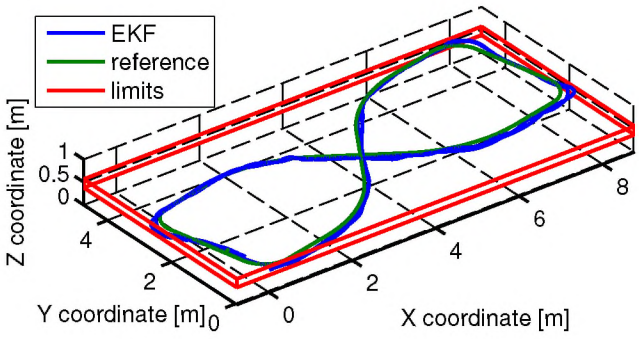

Fig. 9. AR Drone following a horizontal eight-shape trajectory, the yaw heading is kept constant. The quadrotor performs the tracking task smoothly. The maximum speed is reached on the middle of the eight-shape.

following the same eight-shape trajectory to check whether the trajectory control actions are decoupled from the yaw reference. The obtained results were a mean and maximum tracking errors of 0.09 m and 0.28 m correspondingly. The vehicle navigated at a mean and maximum speeds of 0.61 m/s and 1.10 m/s. The similarity between both tests demonstrates that the yaw heading is approximately decoupled from the trajectory tracking.

Fig.	(Pel.) 6	(Pel.) 7	(ARD.) 8	(ARD.) 9
d_{mean} [m]	0.15	0.09	0.10	0.09
d_{max} [m]	0.38	0.27	0.29	0.28
v_{mean} [m/s]	0.42	0.51	0.61	0.61
v_{max} [m/s]	0.70	0.60	1.10	1.10

TABLE I

SUMMARY OF RESULTS OF THE EXPERIMENTAL TESTS: average, d_{mean}, and maximum, d_{max}, tracking error; average, v_{mean}, and maximum, v_{max}, speed during the test

The flights of both quadrotors were mostly anaffected during yaw heading rotation. In the flights exposed in this section the yaw rotation speed was about 25-30 °/s. The experimental data clarify, however, that both vehicles tend to oscillate around the reference trajectory during the yaw rotation, a comparison between Figs. 8 & 9 shows this fact. High yaw rotation speeds do affect the position control, but this is not interesting for many applications.

VI. CONCLUSIONS

In this paper, we presented our experimental work on the design of a multirotor navigation controller that allows for safer multirotor flights. The contribution of this paper is two-fold: first, a discussion on various causes for multirotor navigation controllers malfunctioning is presented. Second, we derived a simple and robuts approach to address these causes natively in our middle-level controller architecture. As a further contribution to the research community, all the code related to this paper is available in two GitHub repositories.

This work is centered around the fact that the state estimation algorithms and the controller can interact in a very negative way when visual odometry estimations are faulty. This motivated our work to identify its causes and design

an approach to overcome the problem. Our experimental work has led us to identify the safety boundaries under which our estimation algorithms work correctly. Then, our controller was designed to natively allow the saturation of the vehicle velocity. Although the speed can be saturated on the trajectory planning module, introducing this feature natively on the controller also allows to fly safely on speed and position control modes, or whenever the middle-level controller is utilized. The work has resulted in an overall increase of the safety and repeatibility of our experimental tests, which in turn allowed us to increase our test's efficiency and gain two awards on the IMAV 2012 competition. The experimental tests on two of our multirotor platforms have shown the robustness of our approach. The small trajectory tracking errors, measured against the EKF estimation, on these tests show that our efforts should be focused on improving our odometry and localization algorithms rather than the controller itself.

As an overall summary, the presented architecture was developed to allow a fast implementation on multiple multirotor vehicles. It permits to test the autonomous navigation software during real flight with a low cost vehicle, such as an AR Drone, and then easily make it portable to a more professional vehicle, such as an Asctec Pelican. The paper highlights the usage of a common drone model, and the issues that had to be solved to attain portability among multiple platforms. As future work, our group is interested on visual localization algorithms that can improve odometry and position measurements to the EKF; and the presented controller architecture will be tested in outdoors adding the GPS measurements for civilian applications research.

ACKNOWLEDGMENT

This work was supported by the Spanish Science and Technology Ministry under the grant CICYT DPI2010-20751-C02-01, and the following institutions through their scholarship grants: CSIC-JAE, CAM and the Chinese Scholarship Council. All the authors are with the Computer Vision Group, www.vision4uav.eu, which belongs to the Centre for Automation and Robotics, joint research center from the Spanish National Research Council (CSIC) and the Polytechnic University of Madrid (UPM).